Nikto Web Server Vulnerability Scanner - Complete Project Guide

Table of Contents

Project Overview

This project demonstrates the use of **Nikto**, a powerful open-source web server vulnerability scanner, on Kali Linux running in Oracle VirtualBox. This guide provides comprehensive coverage from installation to advanced analysis techniques.

Environment:

- Operating System: Kali Linux
- Virtualization: Oracle VirtualBox
- Tool: Nikto Web Scanner
- Target: Web servers and web applications

What is Nikto?

Definition

Nikto is an open-source web server scanner that performs comprehensive tests against web servers for multiple items, including over 6,700 potentially dangerous files/programs, checks for outdated versions of over 1,250 servers, and version-specific problems on over 270 servers.

Purpose

Nikto is designed to:

- Identify security vulnerabilities in web servers
- Detect misconfigurations
- Find outdated software versions
- Discover default files and programs
- Test for common security issues
- Enumerate server information

How Nikto Works

1. **HTTP Requests**: Sends multiple HTTP requests to the target server
2. **Response Analysis**: Analyzes server responses for known vulnerabilities
3. **Pattern Matching**: Compares responses against a database of known issues
4. **Plugin System**: Uses plugins to test for specific vulnerabilities
5. **Reporting**: Generates detailed reports of findings

Installation Guide

Prerequisites

Before installing Nikto, ensure your Kali Linux system is updated.

Step 1: Update System Packages

bash

sudo apt update && sudo apt upgrade -y

Step 2: Install Nikto

Nikto usually comes pre-installed on Kali Linux. To verify or install:

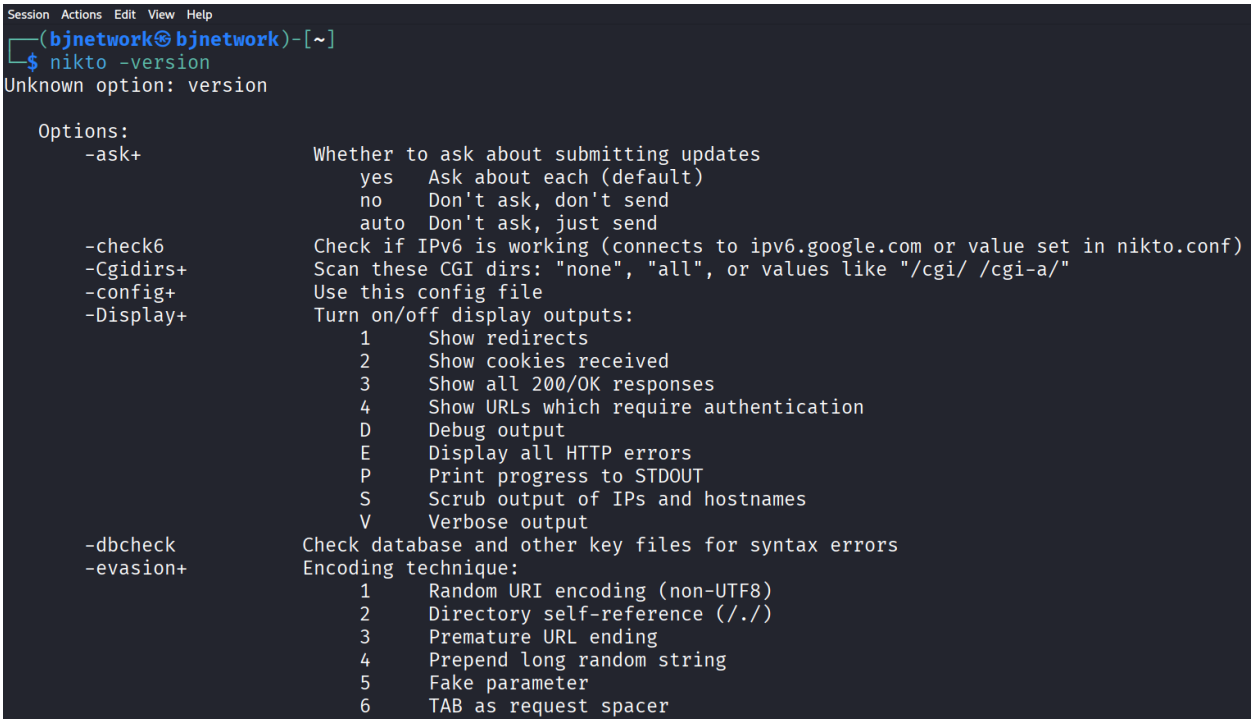
bash

Check if Nikto is installed

nikto -Version

 # If not installed, install it

sudo apt install nikto -y



Step 3: Update Nikto Database

bash

 # Update Nikto's vulnerability database

sudo nikto -update

Step 4: Verify Installation

bash

 # Display help menu

nikto -H

Command Reference

Basic Commands

1. Simple Scan

bash

nikto -h <target_IP_or_domain>

2. Scan with SSL/TLS

bash

nikto -h <target_IP_or_domain> -ssl

3. Scan Specific Port

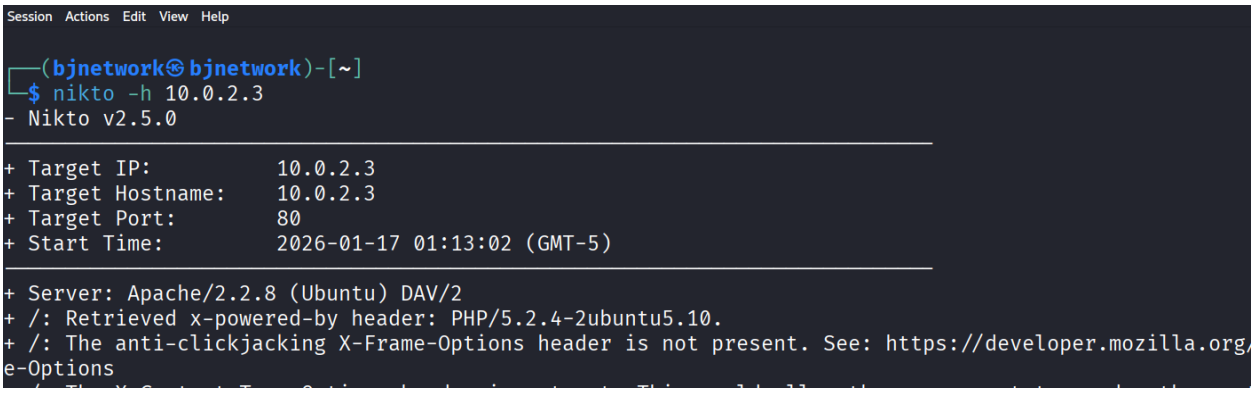
bash

nikto -h <target_IP_or_domain> -p <port_number>

4. Scan Multiple Ports

bash

nikto -h <target_IP_or_domain> -p 80,443,8080



Advanced Commands

5. Save Output to File

bash

HTML format

nikto -h <target> -o scan_results.html -Format html

Text format

nikto -h <target> -o scan_results.txt -Format txt

CSV format

nikto -h <target> -o scan_results.csv -Format csv

XML format

nikto -h <target> -o scan_results.xml -Format xml

6. Scan with Specific Tuning Options

bash

Tuning options:

0 - File Upload

1 - Interesting File / Seen in logs

2 - Misconfiguration / Default File

3 - Information Disclosure

4 - Injection (XSS/Script/HTML)

5 - Remote File Retrieval - Inside Web Root

6 - Denial of Service

7 - Remote File Retrieval - Server Wide

8 - Command Execution / Remote Shell

9 - SQL Injection

a - Authentication Bypass

b - Software Identification

c - Remote Source Inclusion

x - Reverse Tuning Options (exclude)

nikto -h <target> -Tuning 123 # *Test for options 1, 2, and 3*

7. Evasion Techniques

bash

Use evasion techniques to bypass IDS/IPS

nikto -h <target> -evasion 1

Evasion options:

1 - Random URI encoding (non-UTF8)

2 - Directory self-reference (/./)

3 - Premature URL ending

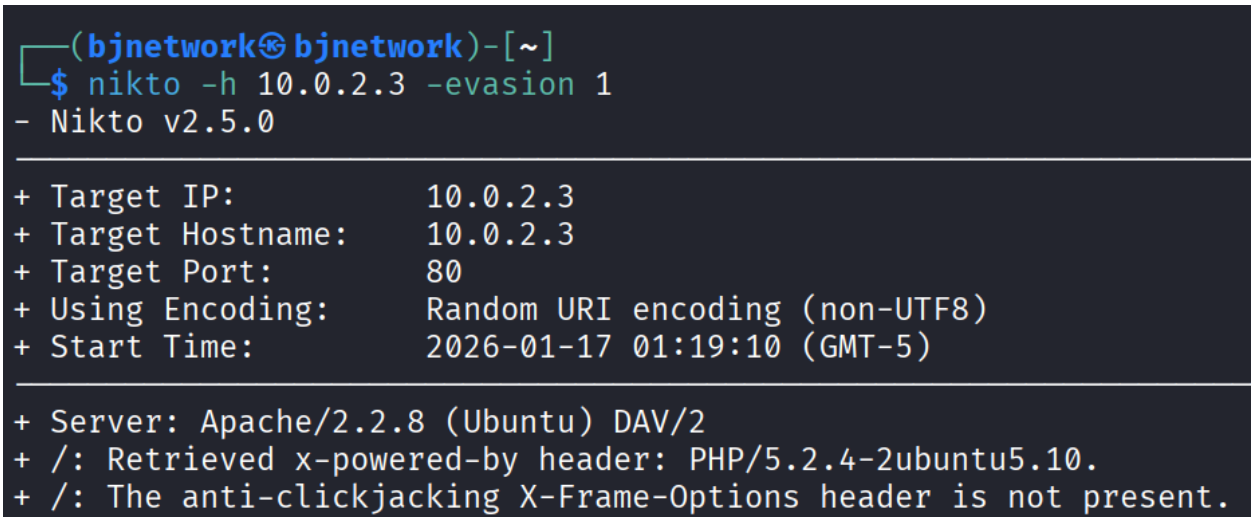
4 - Prepend long random string

5 - Fake parameter

6 - TAB as request spacer

7 - Change the case of the URL

8 - Use Windows directory separator (\)



8. Custom User Agent

bash

nikto -h <target> -useragent "Mozilla/5.0 (Custom Agent)"

9. Follow Redirects

bash

nikto -h <target> -maxtime 30s -maxredirects 5

10. Scan Behind Proxy

bash

nikto -h <target> -useproxy http://proxy_ip:port

11. Disable SSL Certificate Check

bash

nikto -h <target> -ssl -nossl

12. Verbose Output

bash

nikto -h <target> -Display V

Complete Command Examples

bash

Comprehensive scan with output

nikto -h 192.168.1.100 -ssl -p 443 -Tuning 123456789abc -o full_scan.html -Format html

Quick scan with evasion

nikto -h example.com -evasion 1234 -Display V

Multiple targets from file

nikto -h targets.txt -o results.csv -Format csv

Step-by-Step Walkthrough

Phase 1: Pre-Scanning Preparation

Step 1: Set Up Virtual Environment

1. Launch Oracle VirtualBox
2. Start Kali Linux VM
3. Log into your Kali Linux system

Step 2: Verify Network Connectivity

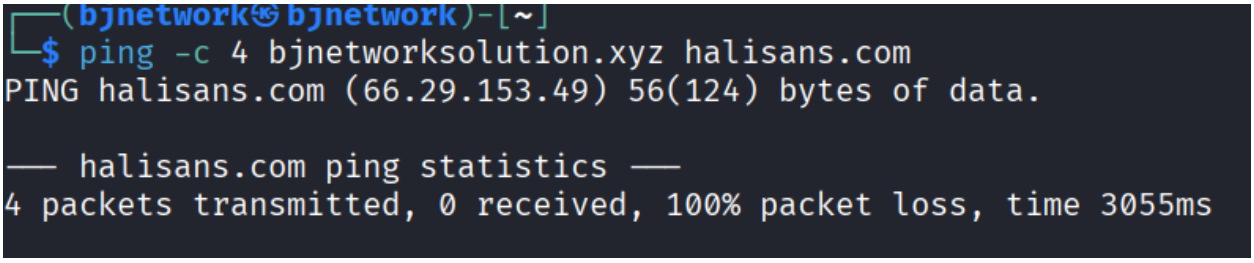
bash

Check network interface

ip addr show

Test internet connectivity

ping -c 4 google.com



Step 3: Identify Target

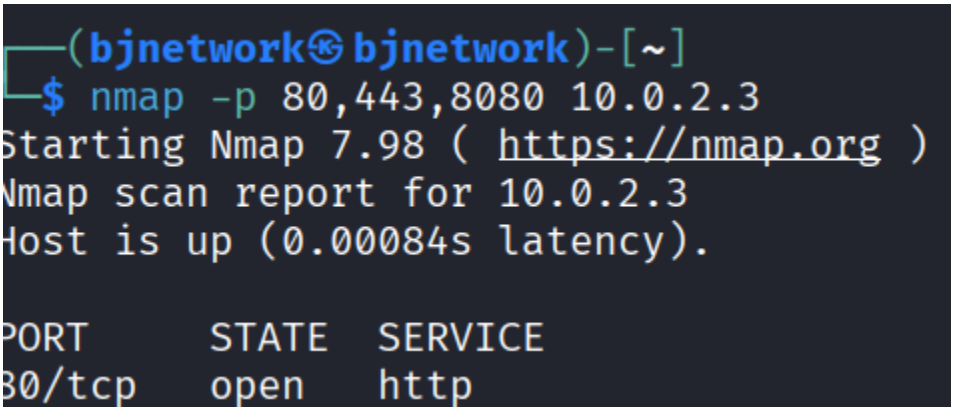
bash

Scan network for web servers (using nmap)

nmap -p 80,443,8080 192.168.1.0/24

Or use a specific target

Example: 192.168.1.100 or scanme.nmap.org



Phase 2: Basic Scanning

Step 4: Perform Initial Reconnaissance

bash

Check if web server is accessible

curl -I http://<target_IP>

Or for HTTPS

curl -Ik https://<target_IP>

Step 5: Run Basic Nikto Scan

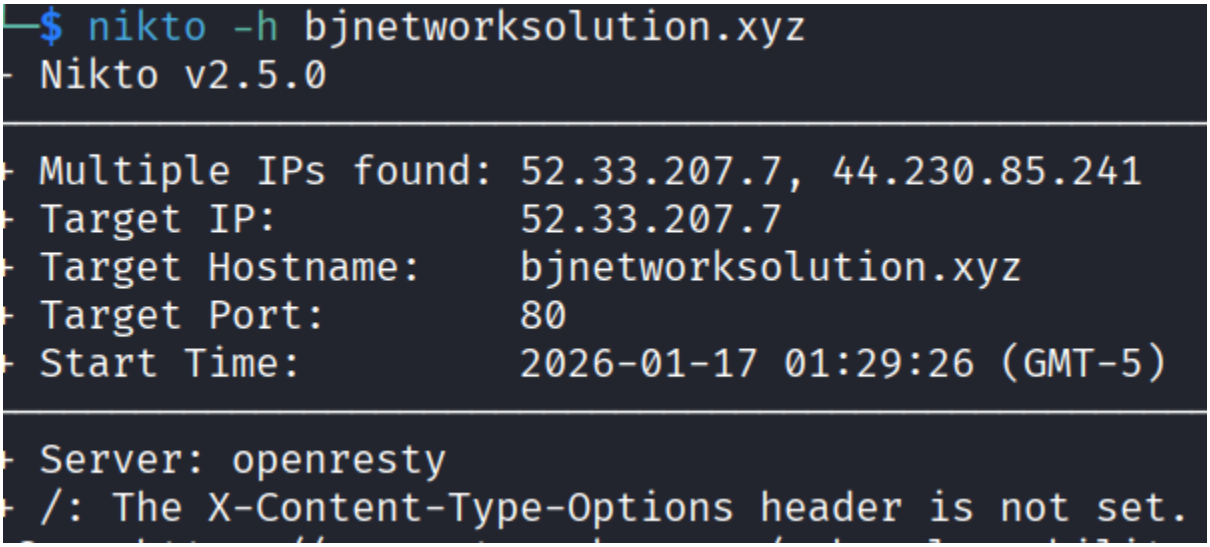
bash

Basic scan

nikto -h <target_IP>

What to observe:

- Server type and version
- Allowed HTTP methods
- Interesting headers
- Detected vulnerabilities



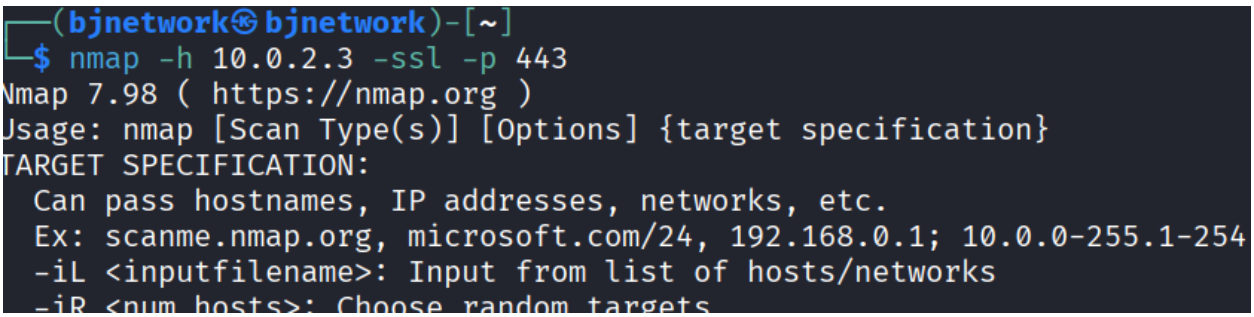
Phase 3: Advanced Scanning

Step 6: SSL/TLS Scan

bash

For HTTPS sites

nikto -h <target_IP> -ssl -p 443



Step 7: Comprehensive Scan with All Tests

bash

Run all tuning options

nikto -h <target_IP> -Tuning 123456789abc -Display V -o comprehensive_scan.html -Format html

Step 8: Targeted Vulnerability Scan

bash

Focus on specific vulnerability types

nikto -h <target_IP> -Tuning 489 -o injection_scan.txt

Phase 4: Results Analysis

Step 9: Review Output Files

bash

View HTML report in browser

firefox comprehensive_scan.html &

View text report

cat injection_scan.txt | less

Step 10: Extract Critical Findings

bash

Filter for critical issues

grep -i "OSVDB\|vulnerable\|exploit" comprehensive_scan.txt

Count vulnerabilities

grep -c "+" comprehensive_scan.txt

Important Information to Look For

During Scanning

1. Server Information Disclosure

- Server type and version (Apache, Nginx, IIS, etc.)
- Operating system information
- Server technologies (PHP, ASP.NET, etc.)

Why it matters: Attackers can use version information to find specific exploits

2. HTTP Methods Allowed

- Dangerous methods: PUT, DELETE, TRACE, CONNECT
- WebDAV methods: PROPFIND, PROPPATCH, MKCOL, COPY, MOVE

Why it matters: Dangerous methods can allow file uploads or server manipulation

3. Security Headers

Look for missing or misconfigured headers:

- X-Frame-Options
- X-XSS-Protection
- X-Content-Type-Options
- Strict-Transport-Security (HSTS)
- Content-Security-Policy (CSP)

Why it matters: Missing security headers expose the site to various attacks

4. Default Files and Directories

- Default installation files
- Admin interfaces (/admin, /phpmyadmin)
- Backup files (.bak, .old, .backup)
- Configuration files exposed

Why it matters: Default files often contain sensitive information or vulnerabilities

5. Outdated Software

- Old server versions
- Outdated CMS versions (WordPress, Joomla, Drupal)
- Deprecated technologies

Why it matters: Outdated software has known vulnerabilities with public exploits

6. Authentication Issues

- Default credentials
- Weak authentication mechanisms
- Authentication bypass possibilities

Why it matters: Weak authentication allows unauthorized access

7. Directory Listings

- Enabled directory browsing
- Exposed file structures

Why it matters: Reveals site structure and potentially sensitive files

8. SSL/TLS Issues

- Weak cipher suites
- SSL certificate problems
- Vulnerable SSL/TLS versions (SSLv2, SSLv3)

Why it matters: Weak encryption can be broken, exposing data

Post-Scan Analysis

What to Analyze After Scanning

1. Severity Classification

Categorize findings by severity:

Critical:

- Remote code execution vulnerabilities

- SQL injection possibilities
- Authentication bypass
- Default credentials

High:

- Cross-site scripting (XSS) vulnerabilities
- Sensitive information disclosure
- Insecure SSL/TLS configuration
- Dangerous HTTP methods enabled

Medium:

- Missing security headers
- Directory listing enabled
- Software version disclosure
- Outdated software versions

Low:

- Informational findings
- Best practice recommendations
- Minor misconfigurations

2. False Positive Verification

Not all findings are actual vulnerabilities. Verify by:

bash

Manual verification with curl

curl -X OPTIONS http://<target> -v

Test specific vulnerability

curl -X TRACE http://<target>

Check for file existence

curl http://<target>/admin/ -I

3. Impact Assessment

For each finding, determine:

- **Exploitability**: How easy is it to exploit?
- **Impact**: What damage could it cause?
- **Affected Systems**: What components are vulnerable?
- **Data at Risk**: What sensitive data could be exposed?

4. Prioritization Matrix

Severity	Exploitability	Priority	Action Timeline
Critical	Easy	P1	Immediate (24h)
Critical	Medium	P1	1-3 days
High	Easy	P2	1 week
High	Medium	P2	2 weeks
Medium	Any	P3	1 month
Low	Any	P4	Backlog

5. Documentation

Create detailed documentation including:

markdown

Vulnerability Report Template

****Finding ID:**** NIKTO-001

Title: Apache Server Version Disclosure

Severity: Medium

CVSS Score: 5.3

Description:

The web server discloses its version in HTTP headers, revealing it's running Apache 2.4.41.

Evidence:

Server: Apache/2.4.41 (Ubuntu)

Impact:

Attackers can identify specific vulnerabilities associated with this version.

Proof of Concept:

curl -I http://192.168.1.100

Remediation:

Configure Apache to hide version information:

- Edit /etc/apache2/conf-available/security.conf
- Set ServerTokens Prod
- Set ServerSignature Off
- Restart Apache: sudo systemctl restart apache2

References:

- CWE-200: Information Exposure
- OSVDB-3233: Apache Version Disclosure

Status: Open

Assigned To: Web Admin Team

Due Date: [Date]

Using Scan Results

1. For Security Assessment

Penetration Testing Workflow:

bash

Step 1: Reconnaissance (Nikto scan completed)

Step 2: Use findings to plan exploitation

Step 3: Exploit vulnerabilities (with permission)

Step 4: Document everything

Step 5: Provide remediation recommendations

2. For Vulnerability Management

- Import results into vulnerability management systems
- Track remediation progress
- Schedule regular scans
- Generate compliance reports

3. For Compliance

Match findings against compliance requirements:

- **PCI DSS**: Quarterly vulnerability scans
- **HIPAA**: Regular security assessments
- **ISO 27001**: Risk assessment requirements
- **GDPR**: Security by design verification

4. For Remediation

Common Remediation Actions:

bash

Disable dangerous HTTP methods (Apache)

Edit /etc/apache2/apache2.conf

```
<Directory />
    <LimitExcept GET POST HEAD>
        Require all denied
    </LimitExcept>
</Directory>
```

Add security headers (Apache)

Edit /etc/apache2/conf-available/security.conf

```
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-XSS-Protection "1; mode=block"
Header always set X-Content-Type-Options "nosniff"
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
```

Disable directory listing

```
Options -Indexes
```

Hide server version

```
ServerTokens Prod
ServerSignature Off
```

5. For Reporting

Generate executive summaries:

markdown

Executive Summary

Scan Date: [Date]

Target: 192.168.1.100

Total Findings: 47

Risk Distribution:

- Critical: 3
- High: 8
- Medium: 21
- Low: 15

Top 3 Risks:

1. Default admin credentials (Critical)
2. SQL injection vulnerability (Critical)
3. Missing security headers (High)

Recommended Actions:

1. Change default credentials immediately
2. Update web application to patch SQL injection
3. Implement security headers
4. Update server software

Advantages and Disadvantages

Advantages

1. Comprehensive Testing

- Tests for over 6,700 potentially dangerous files
- Checks for 1,250+ outdated server versions

- Identifies misconfigurations across 270+ server types

2. Easy to Use

- Simple command-line interface
- Minimal configuration required
- Clear, readable output

3. Free and Open Source

- No licensing costs
- Active community support
- Regular updates and improvements

4. Flexible Output

- Multiple output formats (HTML, XML, CSV, TXT)
- Customizable reporting
- Easy integration with other tools

5. Plugin Architecture

- Extensible through plugins
- Custom tests can be added
- Community-contributed plugins

6. Evasion Capabilities

- Built-in IDS/IPS evasion techniques
- Helps test security controls
- Simulates sophisticated attackers

7. Regular Updates

- Frequent database updates
- New vulnerability checks added
- Keeps pace with emerging threats

8. Cross-Platform

- Works on Linux, Windows, macOS

- Portable
- No special requirements

Disadvantages ❌

1. High False Positive Rate

- Many findings require manual verification
- Generic tests may not apply to all configurations
- Can overwhelm analysts with information

2. Noisy Scanning

- Generates significant server logs
- Easily detected by IDS/IPS
- Not suitable for stealth assessments

3. No Exploitation

- Only identifies vulnerabilities
- Doesn't verify exploitability
- Requires manual validation

4. Limited Scope

- Focuses only on web servers
- Doesn't test application logic deeply
- Misses complex vulnerabilities

5. Slow Performance

- Comprehensive scans take significant time
- Single-threaded by default
- Can impact server performance

6. Outdated Database Entries

- Some checks may be obsolete

- Requires regular updates
- May miss zero-day vulnerabilities

7. No Authentication Testing

- Limited ability to test authenticated areas
- Requires additional tools for complete assessment
- Session handling is basic

8. Legal Risks

- Scanning without permission is illegal
- Can cause service disruption
- May violate terms of service

Security Best Practices

Ethical and Legal Considerations

1. Always Get Permission

✅ DO:

- Scan your own systems

- Get written authorization for client systems

- Use legal test targets (scanme.nmap.org)

- Follow scope of engagement

❌ DON'T:

- Scan systems without permission

- Exceed authorized scope

- Scan production systems during business hours without approval

- Share or sell vulnerability data

2. Responsible Disclosure

If you find vulnerabilities:

1. Contact the organization privately
2. Give them time to patch (typically 90 days)
3. Don't publicly disclose until patched
4. Don't exploit vulnerabilities maliciously

Operational Best Practices

3. Minimize Impact

bash

Reduce scan speed to minimize server load

nikto -h <target> -Pause 2

Limit scan duration

nikto -h <target> -maxtime 600

Scan during off-peak hours

Schedule scans: 2-5 AM local time

4. Keep Tools Updated

bash

Regular update routine

sudo apt update

sudo apt upgrade nikto

nikto -update

Check version

nikto -Version

5. Document Everything

- Maintain scan logs
- Record authorization
- Document findings
- Track remediation

6. Use Controlled Environments

- Test in staging/development first
- Use isolated lab environments
- Avoid production systems when possible

7. Combine with Other Tools

Nikto is most effective when combined with:

bash

Reconnaissance

nmap -sV -sC <target>

Web application scanning

owasp-zap <target>

burpsuite

SSL/TLS testing

sslscan <target>

testssl.sh <target>

Vulnerability verification

searchsploit <software_version>

metasploit

8. Secure Your Scanning System

bash

Update Kali Linux regularly

sudo apt update && sudo apt full-upgrade

Use VPN when scanning

sudo openvpn config.ovpn

Encrypt stored results

gpg -c scan_results.txt

9. Rate Limiting and Throttling

bash

Slow down scans to avoid detection

nikto -h <target> -Pause 3 -maxtime 1800

10. Verify Before Reporting

- Manually verify critical findings
- Test exploitability (in authorized scope)
- Eliminate false positives
- Provide proof of concept

Conclusions and Recommendations

Key Learnings

1. Nikto is a Powerful Reconnaissance Tool

- Excellent for initial web server assessment

- Identifies low-hanging fruit quickly
- Foundation for deeper testing

2. Results Require Interpretation

- Not all findings are exploitable
- Context matters
- Manual verification is essential

3. Part of a Larger Toolkit

- Use with nmap, Burp Suite, OWASP ZAP
- Each tool has strengths and weaknesses
- Comprehensive security requires multiple approaches

4. Regular Scanning is Critical

- New vulnerabilities emerge constantly
- Configuration drift happens
- Continuous monitoring is essential

Recommendations for Different Users

For Students and Beginners

1. **Start with Safe Targets**
    - Use your own lab environment
    - Practice on scanme.nmap.org
    - Set up vulnerable VMs (DVWA, Metasploitable)
2. **Understand Before Scanning**
    - Learn HTTP protocol basics
    - Understand common web vulnerabilities
    - Study OWASP Top 10

3. **Build a Lab**
    - Set up VirtualBox with Kali Linux
    - Deploy vulnerable web applications
    - Practice in a safe environment

For Security Professionals

1. **Integrate into Workflow**
    - Include in penetration testing methodology
    - Automate regular scans
    - Combine with manual testing
2. **Customize and Extend**
    - Create custom plugins
    - Tune for specific environments
    - Build automated reporting
3. **Stay Current**
    - Update regularly
    - Follow security blogs
    - Participate in community

For System Administrators

1. **Proactive Scanning**
    - Scan your own systems regularly
    - Address findings promptly
    - Track remediation
2. **Harden Web Servers**
    - Remove default files
    - Disable unnecessary features
    - Implement security headers
3. **Monitor and Alert**

- Set up IDS/IPS
- Monitor logs for scan activity
- Investigate unauthorized scans

Final Recommendations

Technical Recommendations

1. **Always verify findings manually**
2. **Use multiple scanning tools**
3. **Keep tools and databases updated**
4. **Document everything thoroughly**
5. **Follow responsible disclosure practices**

Process Recommendations

1. **Establish clear scope and authorization**
2. **Schedule scans during maintenance windows**
3. **Create remediation workflows**
4. **Track metrics over time**
5. **Regular training and skill development**

Strategic Recommendations

1. **Implement defense in depth**
2. **Adopt security by design principles**
3. **Build security into SDLC**
4. **Foster security culture**
5. **Continuous improvement mindset**

Analysis Checklist

Critical Items to Look for After Scanning

IMMEDIATE ATTENTION REQUIRED

- Default credentials detected

- Remote code execution vulnerabilities
- SQL injection points identified
- Authentication bypass methods
- Exposed administrative interfaces
- Publicly accessible backup files
- Configuration files exposed
- Dangerous HTTP methods enabled (PUT, DELETE)

🔴 **HIGH PRIORITY**

- Cross-site scripting (XSS) vulnerabilities
- Outdated software with known exploits
- Missing critical security patches
- Weak SSL/TLS configuration
- Information disclosure issues
- Session management problems
- Missing security headers
- Directory traversal vulnerabilities

🟡 **MEDIUM PRIORITY**

- Directory listing enabled
- Server version disclosure
- Unnecessary services enabled
- Verbose error messages
- Insecure cookie settings
- Missing HSTS header
- Open redirects
- Clickjacking vulnerabilities

🟢 LOW PRIORITY / INFORMATIONAL

- Banner information
- Uncommon headers
- Server fingerprinting details
- Best practice recommendations
- Documentation links
- Potential vulnerabilities requiring verification

Project Conclusion

This comprehensive guide to Nikto provides everything needed to effectively use this powerful web vulnerability scanner. Remember:

- **Security is a journey, not a destination**
- **Always act ethically and legally**
- **Continuous learning is essential**
- **Collaboration strengthens security**

Happy (legal and authorized) scanning!